Reevaluating Immune-Inspired Hypermutations Using the Fixed Budget Perspective

Jansen, Thomas; Zarges, Christine

Published in:
IEEE Transactions on Evolutionary Computation

DOI:
10.1109/TEVC.2014.2349160

Publication date:
2014

Citation for published version (APA):
Jansen, T., & Zarges, C. (2014). Reevaluating Immune-Inspired Hypermutations Using the Fixed Budget Perspective. *IEEE Transactions on Evolutionary Computation*, *18*(5), 674-688. https://doi.org/10.1109/TEVC.2014.2349160

Re-Evaluating Immune-Inspired Hypermutations Using the Fixed Budget Perspective

Thomas Jansen and Christine Zarges

***Abstract*—Different studies have theoretically analysed the performance of artificial immune systems in the context of optimisation. It has been pointed out that in comparison with evolutionary algorithms and local search, hypermutations tend to be inferior on typical example functions. These studies have used the expected optimisation time as performance criterion and cannot explain why artificial immune systems are popular in spite of these proven drawbacks. Recently, a different perspective for theoretical analysis has been introduced concentrating on the expected performance within a fixed time frame instead of the expected time needed for optimisation. Using this perspective we re-evaluate the performance of somatic contiguous hypermutations and inverse fitness-proportional hypermutations in comparison with random local search on one well-known example function where random local search is known to be efficient and much more efficient than these hypermutations with respect to the expected optimisation time. We prove that, depending on the choice of the initial search point, hypermutations can by far outperform random local search in a given time frame. This insight helps to explain the success of seemingly inefficient mutation operators in practice. Moreover, we demonstrate how one can benefit from these theoretically obtained insights by designing more efficient hybrid search heuristics.**

***Index Terms*—Computational complexity; Algorithm design and analysis; Heuristic algorithms; Computational intelligence**

I. Introduction

ARTIFICIAL IMMUNE systems are a large class of algorithms that either model or are inspired by the immune system of vertebrates [1]. When used as nature-inspired search heuristics for optimisation they are in direct competition with other such heuristics as evolutionary algorithms [2], ant colony optimisation [3], and simulated annealing [4]. Also other general randomised search heuristics which are not necessarily nature-inspired like tabu search [5] and random local search [6] can be applied to the same optimisation problems. It is therefore natural to ask what kind of search heuristic performs best. Such questions are often examined using typical example or benchmark functions [7].

In comparison to theory of evolutionary computation, theory of artificial immune systems is a rather novel field that has significantly advanced over the last couple of years. In the context of optimisation, research started with the analysis of Markov chain models and convergence in the early 2000's (see [8] for an overview). Since 2008 runtime analysis on different aspects of artificial immune systems have emerged, in particular with respect to working principles of specific operators such as mutation [9]–[12] and ageing [13]–[15]. Initial results for combinatorial optimisation (Vertex Cover [16], Longest Common Subsequence [17]) are available. Very recently first results in the area dynamic optimisation have been presented [15], [18]. For the sake of completeness we remark that theoretical studies in other subareas of artificial immune systems exist, e. g., in the context of classification we refer the reader to the above review article [8] and recent work by Textor et al. [19], [20].

One aspect that sets artificial immune systems apart from other randomised search heuristics is that they employ mutations at a high rate, called somatic hypermutations [21]. There are different variants of such hypermutation operators, among them inversely fitness-proportional mutation (as used in CLONALG [22] and opt-aiNet [23]) and somatic contiguous hypermutation as used in the B-cell algorithm [24].

If one wants to understand and compare the performance of different mutation operators it makes sense to consider them in a minimalistic algorithmic framework to study them in isolation. Following this approach the performance of inversely fitness-proportional mutation and somatic contiguous hypermutation has been analysed on simple example functions and been compared with standard bit mutations which are commonly used in evolutionary algorithms (see [9], [10] for inversely fitness-proportional mutation and [11] for somatic contiguous hypermutation). One of the simplest example functions used for such analytical purposes is known as ONEMAX. It operates on bit strings of fixed length and yields as function value simply the number of 1-bits. Finding its unique global optimum, the all ones bit string, should be a very simple task for any reasonable randomised search heuristic. However, analyses reveal that both kinds of hypermutation perform much worse on ONEMAX than standard bit mutation from evolutionary algorithms or random local search [9], [11]. These results appear to suggest that using hypermutation is not a very good idea and are in apparent contrast to the popularity and actual usefulness of artificial immune systems as optimisers. The contradiction can be explained by the analytical perspective that is adopted in such studies.

When analysing randomised search heuristics one often uses the number of evaluations of the objective function as performance measure until a solution of sufficient quality is found [25]. If 'sufficient quality' is defined as some approximation of an optimal solution this number is called the

Thomas Jansen is with the Department of Computer Science, Aberystwyth University, UK.

Christine Zarges is with the School of Computer Science, University of Birmingham, UK.

Manuscript first submitted to IEEE, December 2013; resubmitted in April 2014, June 2014 and July 2014; accepted in July 2014.

approximation time, if one waits for an optimal solution to be found it is called optimisation time. It has recently been pointed out [26], [27] that this performance measure is in strange contradiction to the way randomised search heuristics are usually applied. Clearly, in practice, the optimal value is not known and thus one cannot stop once it is reached. Most often a randomised search heuristic is stopped after a pre-specified number of steps. Taking this into account the analytical perspective of fixed budget computations has been introduced [26]. This new measure allows to perform more realistic, detailed, and practically relevant analyses. It is, however, technically more challenging than analysing the optimisation time. To date, results for the optimisation time can only be used to infer fixed budget results if additionally tail bounds for the optimisation time are known [28]. Recently, a first fixed budget analysis for a combinatorial optimisation problem, namely the traveling salesperson problem, has been presented [29].

We use this perspective of fixed budget computations to re-evaluate the performance of inversely fitness-proportional and somatic contiguous hypermutation. We perform an analysis for ONEMAX and compare both with random local search since this very simple search heuristic is for this example function even faster than standard bit mutations from evolutionary algorithms: The $(1+1)$ EA using standard bit mutations with probability $1/n$ has expected optimisation time $en\ln n - \Theta(n)$ [30] while random local search (which for ONEMAX corresponds to the well-known Coupon Collector process) only requires $n\ln n + \Theta(n)$ steps [31]. We prove that in comparison with this strong competitor hypermutations from artificial immune systems actually perform very well. They can have a very large advantage in the beginning of the run and only losing out later when approaching an optimal solution. This explains their very good performance in experiments that concentrate on not too long runs. We perform the analysis for different starting points, specific starting points as well as randomly selected ones. Moreover, we accompany our findings with the results of experiments to demonstrate how our bounds on the expected performance compare with empirical results. Finally, we turn our theoretically gained insights into practice by using them to devise useful hybrid strategies.

In the next section we define the analytical framework and the different mutation operators as well as the algorithmic framework we apply. We also state the known results for random local search there. Section III contains analytical results for somatic contiguous hypermutation. Section IV contains analytical results for inversely fitness-proportional mutation. We provide empirical supplements to our theoretical bounds in Section V. We demonstrate that theoretical analysis and insights gained in theoretical studies are of practical importance in Section VI. Using our results on the benefits of the different mutation operators we design hybrid algorithms that benefit from combining both mutation operators in a way that is informed by theoretically obtained results. We conclude and outline possible future research directions in Section VII.

II. ANALYTICAL FRAMEWORK

We compare two different hypermutation operators with random local search on the well-known example function ONEMAX. For the sake of clarity we give a formal definition for ONEMAX. For a bit string $x \in \{0,1\}^n$ of length n we denote its bits by $x[0]$, $x[1]$, …, $x[n-1]$.

Definition 1. *The function* ONEMAX$\colon \{0,1\}^n \to \mathbb{N}_0$ *is defined by* ONEMAX$(x) = \sum_{i=0}^{n-1} x[i]$ *for all* $x \in \{0,1\}^n$.

We choose to embed the different mutation operators into a minimalistic algorithmic framework. The search is based on a single point in the search space and works in rounds. In each round a new search point is created by means of mutation. The function values of the two search points are compared and the new search point replaces the current one if its function value is not worse. Note that random local search and the (1+1) evolutionary algorithm [32] both fit this framework while simulated annealing does not (due to the less simple rule for replacing the current search point). We formally define the framework as Algorithm 1.

Input : number of function evaluations b
Output: x_t, $f(x_t)$
1 Set $t := 0$;
2 Select $x_t \in \{0,1\}^n$;
3 **while** $t + 1 < b$ **do**
4 $\quad y :=$ mutate(x_t);
5 $\quad$ **if** $f(y) \geq f(x_t)$ **then**
6 $\quad\quad x_{t+1} := y$
7 $\quad$ **else**
8 $\quad\quad x_{t+1} := x_t$
9 $\quad$ **end**
10 $\quad t := t + 1$;
11 **end**

Algorithm 1: Algorithmic Framework

The algorithmic framework can be instantiated by defining the choice of the initial search point in line 2 and the mutation in line 4. When the mutation operator M is used we denote the search point x_t as $x_t^{(M)}$. Using this notation we define three mutation operators: random local search (Algorithm 2), somatic contiguous hypermutations (Algorithm 3) and inverse fitness-proportional hypermutations (Algorithm 4). We define the algorithms RLS, CHM and CLONALG by using the algorithmic framework (Algorithm 1).

Input : bit string $x = x[0]x[1]\cdots x[n-1]$
Output: bit string $y = y[0]y[1]\cdots y[n-1]$
1 $y := x$;
2 Select $p \in \{0,1,\ldots,n-1\}$ uniformly at random;
3 $y[p] := 1 - y[p]$;

Algorithm 2: Mutation RLS

We consider the performance of the three algorithms RLS, CHM, and CLONALG on ONEMAX. We do this considering their performance in the fixed budget computation model. To distinguish the three algorithms we denote x_t as $x_t^{(\text{RLS})}$, $x_t^{(\text{CHM})}$,

Input : bit string $x = x[0]x[1]\cdots x[n-1]$
Output: bit string $y = y[0]y[1]\cdots y[n-1]$
1 $y := x$;
2 Select $p \in \{0, 1, \ldots, n-1\}$ uniformly at random;
3 Select $l \in \{0, 1, \ldots, n\}$ uniformly at random;
4 **foreach** $i \in \{0, 1, \ldots, l-1\}$ **do**
5 | $y[(p+i) \bmod n] := 1 - y[(p+i) \bmod n]$
6 **end**

Algorithm 3: Mutation CHM

Input : bit string $x = x[0]x[1]\cdots x[n-1]$, $\rho \in \mathbb{R}^+$
Output: bit string $y = y[0]y[1]\cdots y[n-1]$
1 $y := x$;
2 $v := f(y)/n$ #normalise fitness in [0,1];
3 **foreach** $i \in \{0, 1, \ldots, n-1\}$ *independently* **do**
4 | With probability $e^{-\rho \cdot v(y)}$ set $y[i] := 1 - y[i]$.
5 **end**

Algorithm 4: Mutation CLONALG

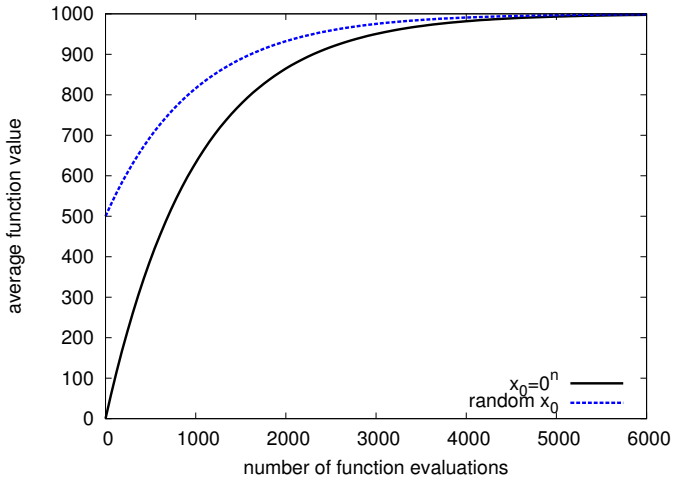

Fig. 1. Visualisation of $\mathrm{E}\left(\textsc{OneMax}\left(x_b^{(\mathrm{RLS})}\right)\right)$ for $n = 1000$, initialisation in 0^n and random initialisation.

$x_t^{(\mathrm{CLONALG})}$ depending on the mutation operator used. In the fixed budget computation model we analyse $\mathrm{E}\left(x_b^{(A)}\right)$ for a fixed number of steps b and $A \in \{\mathrm{RLS}, \mathrm{CHM}, \mathrm{CLONALG}\}$.

We use three different starting points in the initialisation (line 2 in Algorithm 1). We consider deterministic initialisation in the all zero bit string, 0^n, deterministic initialisation in the bit string that consists of $n/2$ 1-bits followed by $n/2$ 0-bits, $1^{n/2}0^{n/2}$, and random initialisation in a bit string $x \in \{0,1\}^n$ selected uniformly at random. For the sake of convenience we assume n to be even. Note that the expected number of 1-bits in a randomly selected x equals $n/2$. Since RLS and CLONALG are not sensitive to the positions of the 1-bits, in expectation, there are no differences in their performance for a random bit string and $1^{n/2}0^{n/2}$. For CHM, many consecutive 0-bits are beneficial. Thus, $1^{n/2}0^{n/2}$ is a best case for CHM under all bit strings with $n/2$ 1- and $n/2$ 0-bits.

For RLS, results are already known. We cite the relevant results from [27]. Note that the result for $x = 1^{n/2}0^{n/2}$ is not contained in [27] but a direct consequence of the fact that RLS is oblivious to reordering of the 1-bits as long as their number is not changed.

Theorem 1 (Theorems 4 and 5 from [27])**.** *With* $x_0^{(RLS)} = 0^n$, $\mathrm{E}\left(\textsc{OneMax}\left(x_b^{(RLS)}\right)\right) = n \cdot \left(1 - (1-1/n)^b\right)$ *holds for all budgets* $b \in \mathbb{N}_0$.

With $x_0^{(RLS)} = 1^{n/2}0^{n/2}$ *and* $x_0^{(RLS)} \in \{0,1\}^n$ *selected uniformly at random,* $\mathrm{E}\left(\textsc{OneMax}\left(x_b^{(RLS)}\right)\right) = (n/2) + (n/2) \cdot \left(1 - (1-1/n)^b\right)$ *holds for all budgets* $b \in \mathbb{N}_0$.

Both statements can be proven by straightforward calculations. We refer to [27] for details and depict both function for $n = 1000$ in Figure 1 for illustration.

III. Somatic Contiguous Hypermutations

A. CHM on OneMax starting in 0^n

We start with a lower bound on the expected function value. The expected progress for CHM heavily depends on the distribution of 0- and 1-bits over the bit string. Thus, already after a few steps deriving the expected progress becomes very involved. Examining a few steps is sufficient to make our point, so we restrict ourselves to the first two iterations, only. We show that we make progress in order of $\Theta(n)$ in the very beginning. Since the step size of RLS is 1 it takes at least $\Omega(n)$ steps until RLS can catch up. Later in the run, however, RLS is clearly faster than CHM so that the expected function value for RLS becomes larger than that for CHM.

Theorem 2. *Let* $x_0^{(CHM)} = 0^n$. *For all* $b \in \mathbb{N} \setminus \{1\}$, $\mathrm{E}\left(\textsc{OneMax}\left(x_b^{(CHM)}\right)\right) \geq (13/24)n$ *holds.*

Proof. Let $F_i := \textsc{OneMax}(x_i)$ for all $i \in \{0, 1, \ldots, b\}$. Let $\Delta_i := F_i - F_{i-1}$ for all $i \in \{1, 2, \ldots, b\}$. We see that

$$F_t = \sum_{i=1}^{t} \Delta_i$$

holds for all $t \in \{1, 2, \ldots, b\}$. Due to the selection we have $F_i \geq F_{i-1}$ for all $i \in \{1, 2, \ldots, b\}$. Thus, for a lower bound on $\mathrm{E}(F_t)$ for not too large t it suffices to consider $\mathrm{E}(\Delta_i)$ for very small i.

We consider the situation with $x_0 = 0^n$ so that $F_0 = 0$ holds. The very first mutation is guaranteed to be accepted and the function value of the new search point equals the number of mutated bits. Thus, we have

$$\mathrm{E}(\Delta_1) = \sum_{l=0}^{n} l \cdot \frac{1}{n+1} = \frac{n}{2}$$

and can already conclude that $\mathrm{E}(F_i) \geq n/2$ holds for all $i \geq 1$.

For a lower bound on $\mathrm{E}(\Delta_2)$ we consider all mutations that only affect 0-bits in x_1. Since the number of 0-bits equals $n - l$ if l is the length of the mutated region in the first mutation we have i 0-bits with probability $1/(n+1)$ for all $i \in \{0, 1, \ldots, n\}$. Given i 0-bits we mutate l of these if in the mutation we select l as length of the mutated region and any position p such that all l bits starting at p are 0-bits. There are

$i+1-l$ such positions for $l \leq i$ and no such positions for larger l. Therefore, we have

$$\mathrm{E}(\Delta_2) \geq \sum_{i=1}^{n} \frac{1}{n+1} \cdot \sum_{l=1}^{i} l \cdot \frac{1}{n+1} \cdot \frac{i+1-l}{n} = \frac{(n+3)(n+2)}{24(n+1)} > \frac{n}{24}$$

and conclude that $F_t \geq 13n/24$ holds for all $t \geq 2$. □

For the upper bound we observe that the expected progress decreases considerably once a sufficiently large number of 1-bits has been collected. Recall that the expected progress for CHM heavily depends on the distribution of the 0-bits over the bit string. We consider the best case, i. e., all remaining 0-bits are consecutive, and show that even in this case the expected progress is too small to keep up with RLS.

Theorem 3. *Let $x_0^{(CHM)} = 0^n$. For all $b \in \mathbb{N}_0$ and all constants $\varepsilon \in (0,1/2)$, $\delta > 2/3$*

$$\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(CHM)}\right)\Big) \leq n - n^{\varepsilon} + \frac{\delta b}{n^{2-3\varepsilon}}$$

holds.

Proof. We want to prove an upper bound on $\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CHM})}\right)\Big)$. Consider some $x_t^{(\mathrm{CHM})}$ for an arbitrary $t \in \mathbb{N}_0$. The expected increase in function value in one generation, i. e., $\mathrm{E}\Big(\textsc{OneMax}\left(x_{t+1}^{(\mathrm{CHM})}\right) - \textsc{OneMax}\left(x_t^{(\mathrm{CHM})}\right) \mid x_t^{(\mathrm{CHM})}\Big)$, is largest when the 1-bits in x_t are all in a single consecutive block. We can therefore restrict our interest to bit strings $x_t^{(\mathrm{CHM})}$ of the form 1^i0^{n-i} with $i = \textsc{OneMax}\left(x_t^{(\mathrm{CHM})}\right)$ when bounding the expected function value.

The proof is carried out in two steps. In the first step we show that

$$\begin{aligned}&\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CHM})}\right) \mid \textsc{OneMax}\left(x_{b-1}^{(\mathrm{CHM})}\right) = j-k\Big)\\ &\leq \mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CHM})}\right) \mid \textsc{OneMax}\left(x_{b-1}^{(\mathrm{CHM})}\right) = j\Big) \quad (*)\end{aligned}$$

holds for any $b \in \mathbb{N}_0$ $j \in \{0,1,\ldots,n\}$ and $k \in \{0,1,\ldots,j\}$. We use this to bound the expected function value after b steps by proving an upper bound on $\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CHM})}\right) \mid \textsc{OneMax}\left(x_0^{(\mathrm{CHM})}\right) = n - n^{\varepsilon}\Big)$ instead of $\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CHM})}\right) \mid \textsc{OneMax}\left(x_0^{(\mathrm{CHM})}\right) = 0\Big)$.

For this it suffices to prove

$$\begin{aligned}&\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CHM})}\right) \mid \textsc{OneMax}\left(x_{b-1}^{(\mathrm{CHM})}\right) = j-1\Big)\\ &\quad\leq \mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CHM})}\right) \mid \textsc{OneMax}\left(x_{b-1}^{(\mathrm{CHM})}\right) = j\Big)\end{aligned}$$

for arbitrary $b \in \mathbb{N}_0$ and $j \in \{1,2,\ldots,n\}$ since the original claim then follows by induction. Remember that we only consider bit strings of the form 1^j0^{n-j}. We consider an arbitrary but fixed mutation for $x = 1^j0^{n-j}$ and $x' = 1^{j-1}0^{n-j+1}$ and are interested in the expected function values of the resulting offspring. We can ignore mutations that are not accepted for both bit strings since they do not contribute to the function values. We can ignore mutations that do not affect the only differing bit since they have identical effects for both x and x'. We can ignore mutations that affect the only differing bit and are only accepted for x': since they are not accepted for x we conclude that the difference in function value between x and its offspring was negative, -1 or smaller. The difference in function value between the offspring of x and x' equals 1 since they differ in exactly one bit. Thus, the difference in function value between x' and its offspring equals 0 and has no contribution to the expected change in function value. The remaining mutations occur with a probability $p < 1$. Their expected additional contribution in x' in comparison to their contribution in x is bounded by p. We conclude that

$$\begin{aligned}&\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CHM})}\right) \mid x_{b-1} = x'\Big)\\ &\qquad\leq \mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CHM})}\right) \mid x_{b-1} = x\Big)\end{aligned}$$

holds since the x contains one 1-bit more and this difference cannot be made up for by the contribution of p.

We now consider the situation for $\textsc{OneMax}\left(x_t^{(\mathrm{CHM})}\right) \geq n - n^{\varepsilon}$. For bit strings of the form 1^i0^{n-i} the expected increase in function value in one generation is strictly decreasing with increasing i. Using the notation from the proof of Theorem 2 we prove a bound on $\mathrm{E}(F_i - F_{i-1} \mid F_{i-1} \geq n - (1+\gamma)n^{\varepsilon})$ for a bit string $x_{i-1} = 1^{F_{i-1}}0^{n-F_{i-1}}$ and a constant $\gamma > 0$ since this yields an upper bound as $(*)$ shows.

The number of 1-bits can be increased in a single mutation by $z \in \{1,2,\ldots,n-F_{i-1}\}$. Consider one fixed value $z \in \{1,2,\ldots,n-F_{i-1}\}$ to see how this can be achieved. The mutation flips exactly $z+j$ 0-bits and j 1-bits for some $j \in \{0,1,2,\ldots,\min\{F_{i-1}, n-F_{i-1}-z\}\}$. Note that exactly $z+2j$ bits flip so that $l = z+2j$ is the only matching choice for the random length of the mutating block. In the case that only 0-bits flip ($j = 0$) there are $n-F_{i-1}-z+1$ possible positions for the beginning of the mutating block. Now consider the case $j > 0$. If not all 0-bits flip ($z+j < n-F_{i-1}$) there can be at most two positions for the beginning of the mutating block, either before the block of 0-bits or within it. Thus, this subcase contributes at most $2(\min\{F_{i-1}, n-F_{i-1}-z\}-1) = 2(n-F_{i-1}-z)-2$ possible positions for the beginning of the mutating block (where the inequality holds since we have $F_{i-1} \gg n-F_{i-1}$). Finally, we consider the case that all 0-bits are flipped. Since we have j flipping 1-bits there are at most $j+1 \leq n-F_{i-1}-z+1$ possible positions for the beginning of the block since the number of mutated 1-bits preceding the block of 0-bits is between j and 0. Together this yields

$$\begin{aligned}&\mathrm{E}(F_i - F_{i-1} \mid F_{i-1} \geq n - (1+\gamma)n^{\varepsilon})\\ &\leq \sum_{z=1}^{n-F_{i-1}} z\left(\frac{(n-F_{i-1}-z+1)+(2(n-F_{i-1}-z)-2)}{n\cdot(n+1)} + \frac{(n-F_{i-1}-z+1)}{n\cdot(n+1)}\right)\\ &= \frac{2\cdot(n-F_{i-1}-1)\cdot(n-F_{i-1})\cdot(n-F_{i-1}+1)}{3n(n+1)}\\ &< \frac{2(n-F_{i-1})^3}{3n^2}.\end{aligned}$$

We make use of the fact that the improvement decreases monotonically with an increasing number of 1-bits and obtain $\mathrm{E}(F_i - F_{i-1} \mid F_{i-1} \geq n-(1+\gamma)n^\varepsilon) < (2/3)\cdot(1+\gamma)^3\cdot n^{3\varepsilon-2}$. We set $\gamma := (3\delta/2)^{1/3}-1$ which is a valid choice since $\delta > 2/3$ implies $(3\delta/2)^{1/3}-1 > 0$. We now have $\mathrm{E}(F_i - F_{i-1} \mid F_{i-1} \geq n-(1+\gamma)n^\varepsilon) < \delta n^{3\varepsilon-2}$ and conclude that for all points of time t where $\textsc{OneMax}\left(x_t^{(\mathrm{CHM})}\right) \geq n-(1+\gamma)n^\varepsilon$ holds we have that the expected increase in function value is bounded by $\delta/n^{2-3\varepsilon}$. Together this yields $\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CHM})}\right)\Big) \leq n-n^\varepsilon+\delta b/n^{2-3\varepsilon}$ as claimed. □

We summarise what we have in the following corollary.

Corollary 1. *Let $x_0^{(CHM)} = 0^n$. For all $b \in \{1, 2, \ldots, \lfloor(13/24)n\rfloor\}$ the expected function value after b function evaluations is larger for CHM than for RLS.*

For all $\varepsilon \in (0,1/2)$ and all $b \in \{\lceil(1-\varepsilon^2)n\ln(n)\rceil, \lceil(1-\varepsilon^2)n\ln(n)\rceil+1, \ldots, \lfloor(3/2)n^{2-2\varepsilon}\rfloor\}$ the expected function value after b function evaluations is larger for RLS than for CHM.

Proof. We have

$$\mathrm{E}\Big(\textsc{OneMax}\left(x_1^{(\mathrm{CHM})}\right)\Big) = n/2 > 1 = \mathrm{E}\Big(\textsc{OneMax}\left(x_1^{(\mathrm{RLS})}\right)\Big)$$

and for all $b \in \{2,3,\ldots,\lfloor(13/24)n\rfloor\}$ we have $\mathrm{E}\big(\textsc{OneMax}\left(x_b^{\mathrm{CHM}}\right)\big) \geq (13/24)n$ (Theorem 2). For RLS we know that the increase in function value that can be achieved in one step is bounded by 1 since only one bit is flipped. This implies

$$\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{RLS})}\right)\Big) \leq b$$

in general and $\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{RLS})}\right)\Big) \leq (13/24)n$ for $b \leq (13/24)n$ in particular.

For $b = \lceil(1-\varepsilon^2)n\ln(n)\rceil$ we have

$$\begin{aligned}\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{RLS})}\right)\Big) &= n\cdot\left(1-\left(1-\frac{1}{n}\right)^{\lceil(1-\varepsilon^2)n\ln(n)\rceil}\right)\\ &\geq n\cdot\left(1-\left(1-\frac{1}{n}\right)^{(1-\varepsilon^2)n\ln(n)}\right)\\ &> n\cdot\left(1-\left(\frac{1}{n}\right)^{1-\varepsilon^2}\right)\\ &= n-n^{\varepsilon^2}.\end{aligned}$$

Even for $b = \lfloor(3/2)n^{2-2\varepsilon}\rfloor\}$ we have $\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CHM})}\right)\Big) \leq n-n^\varepsilon+1 < n-n^{\varepsilon^2}$. □

B. CHM on OneMax starting in $1^{n/2}0^{n/2}$

We consider initialisation $1^{n/2}0^{n/2}$ and see that things are not that different to initialisation in 0^n. As in the previous section we start with a lower bound and consider the expected function value after the first iteration (cf. Theorem 2). We see that we still make large progress in the beginning since all 0-bits are consecutive.

Theorem 4. *Let $x_0^{(CHM)} = 1^{n/2}0^{n/2}$. For all $b \in \mathbb{N}$, $\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(CHM)}\right)\Big) \geq (2/3)n-1$ holds.*

Proof. Using the notation from Theorem 2 we have $F_0 = n/2$ and

$$\begin{aligned}\mathrm{E}(\Delta_1) &= \left(\sum_{p=1}^{n/2}\sum_{l=1}^{(n/2)-p+1}\frac{l}{n(n+1)}\right)\\ &+\left(\sum_{p=1}^{(n/2)-1}\sum_{l=(n/2)+2-p}^{n-2p+1}\frac{n+2-2p-l}{n(n+1)}\right)\\ &+\left(\sum_{p=(n/2)+2}^{n}\sum_{l=2n-2p+3}^{n-1}\frac{l-n+p-1}{n(n+1)}\right)\\ &= \frac{4n^2-9n+14}{24n+24} > \frac{n}{6}-1\end{aligned}$$

and conclude that $F_t \geq (n/2)+(n/6)-1 = (2/3)n-1$ holds for all $t \geq 1$. □

For the upper bound we can use exactly the same argumentation as in Theorem 3.

Theorem 5. *Let $x_0^{(CHM)} = 1^{n/2}0^{n/2}$. For all $b \in \mathbb{N}_0$ and all constants $\varepsilon \in (0,1/2)$, $\delta > 2/3$*

$$\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(CHM)}\right)\Big) \leq n-n^\varepsilon+\frac{\delta b}{n^{2-3\varepsilon}}$$

holds.

Proof. Reconsidering the proof of Theorem 3 we see that the statement holds not only for $x_0^{(\mathrm{CHM})} = 0^n$ but for all $x_0^{(\mathrm{CHM})}$ with $\textsc{OneMax}\left(x_0^{(\mathrm{CHM})}\right) \leq n-n^\varepsilon$. This implies the statement here. □

We again summarise what we have in the following corollary.

Corollary 2. *Let $x_0^{(CHM)} = x_0^{(RLS)} = 1^{n/2}0^{n/2}$.*

For all $b \in \{1, 2, \ldots, \lfloor n/6\rfloor-1\}$ the expected function value after b function evaluations is larger for CHM than for RLS.

For all $\varepsilon \in (0,1/2)$ and all $b \in \{\lceil(1-\varepsilon^2)n\ln(n)\rceil, \lceil(1-\varepsilon^2)n\ln(n)\rceil+1, \ldots, \lfloor(3/2)n^{2-2\varepsilon}\rfloor\}$ the expected function value after b function evaluations is larger for RLS than for CHM.

Proof. In comparison to the proof of Corollary 1 the only change is in the initial function value. In the first mutation the expected function value for CHM increases by at least $\lfloor n/6\rfloor-1$ and by less than 1 for RLS. This proves the first part of the statement. The calculations for the second part remain largely unchanged. □

C. CHM on OneMax with random starting point

Things change if we initialise with a random starting point since now 0-bits are not necessarily consecutive. However, in

the first step CHM has a good chance of making a significant step towards the optimum, a progress of order $\Theta(\sqrt{n})$. Since the step size of RLS is 1 it takes at least $\Omega(\sqrt{n})$ steps until RLS can catch up. Later in the run, however, RLS is clearly faster than CHM so that the expected function value for RLS becomes larger than that for CHM.

Theorem 6. *Let $x_0^{(CHM)}$ and $x_0^{(RLS)}$ be selected uniformly at random. There is a $b_1 \in \mathbb{N}$ with $b_1 \in \Theta(\sqrt{n})$ such that the expected function value after b function evaluations is larger for CHM than for RLS for any $1 \leq b \leq b_1$.*

There is a $b_2 \in \mathbb{N}$ with $b_2 \in \Theta(n)$ such that the expected function value after b function evaluations is larger for RLS than for CHM for any $b \geq b_2$.

Proof. We know $\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{RLS})}\right)\Big)$ for any b exactly from Theorem 1 and see that $\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{RLS})}\right)\Big) \leq (n/2) + b/2$ holds.

For CHM we have $\mathrm{E}\Big(\textsc{OneMax}\left(x_0^{(\mathrm{CHM})}\right)\Big) = n/2$ and consider the first step. This first step flips a sequence of s bits. The number of 0-bits Z in this sequence is binomially distributed with parameters s and $1/2$. We know that $\Pr(Z = k)$ is maximal for $k \in \{s/2, (s+1)/2\}$ and that the maximum equals $c/\sqrt{s}$ for some constant $c > 0$ [33]. We conclude that $\Pr(Z \geq (s/2) + (1/(4c))\sqrt{s}) \geq (1/2) - (1/(4c))\sqrt{s} \cdot c/\sqrt{s} \geq 1/4$ holds. Therefore, flipping this sequence increases the function value by $\Omega(\sqrt{s})$ with probability at least $1/4$. We have $s = \Omega(n)$ with probability $\Omega(1)$. We conclude that the expected increase in function value is $\Omega(\sqrt{n})$ so that $\mathrm{E}\Big(\textsc{OneMax}\left(x_1^{(\mathrm{CHM})}\right)\Big) = (n/2) + \Omega(\sqrt{n})$ holds and the first claim follows.

When the number of 0-bits sinks below the number of 1-bits the probability to have sequences of bits of length s where the number of 0-bits exceeds the number of 1-bits decreases. When the number of 1-bits is by $\Theta(n)$ larger it becomes exponentially small in s. This is the case because we start with a bit string that is selected uniformly at random, because the application of contiguous hypermutations does not change the uniform distribution and because the selection depends only on the number of 1-bits in a bit string. Therefore the current bit string is distributed uniformly at random among all bit strings with an equal number of 1-bits.

The only non-obvious of the above three reasons is the statement about contiguous hypermutations not changing the uniform distribution. This holds in fact for any mutation operator with the property that mutating from x to y has the same probability as mutating from y to x (see the proof of Theorem 5.16 (page 120) in [25] for a proof). For contiguous hypermutations it is the case since a mutation that leads from x to y is characterised by a choice of position p and length l and the same pair of values for p and l characterises a mutation leading from y to x. Thus, for a fixed number of 0-bits z the current bit string is any one of them with equal probability $1/\binom{n}{z}$. Therefore, with $z = (n/2) - \Omega(n)$ the probability to have a sequence of s bits where the number of 0-bits is larger than the number of 1-bits is $e^{-\Omega(s)}$. This implies a bound on the expected increase in function value of $O\big(z \cdot \max\{s^3/\left(n^2 e^s\right) \mid s \in \{1, 2, \ldots, n\}\}\big) = O\big(z/n^2\big)$ where z denotes the total number of remaining 0-bits. The term s^3/n^2 stems from the expected increase in function for a block of up to s 0-bits. The term e^{-s} stems from the decreasing probability of having an appropriate block. The factor z takes care of the number of independent such blocks we may have. Obviously, for not very small s using a factor of z overestimates considerably. After t steps (with $t = \Theta(n)$ sufficiently large) where the expected increase in function value for RLS is $\Theta(1)$ and the expected increase in function value for CHM is $O(1/n)$ the function value for RLS is larger than that of CHM with probability very close to 1. This implies the second statement. □

We see that random initialisation is the only case where classical run time analysis provides an accurate picture of the performance of CHM and RLS.

IV. Inversely Fitness-Proportional Hypermutations

We now consider the mutation operator from CLONALG in the very same way as CHM in the previous section, i. e., with the same initialisation schemes for the sake of consistency. We see that in contrast to CHM, CLONALG mutation is oblivious with respect to the distribution of 0- and 1-bits. We therefore expect to see similar results for random initialisation and initialisation in $1^{n/2}0^{n/2}$. However, it is important to note that the mutation rate depends on the current function value and thus, changes over time. This makes the analysis of this operator a challenging task. We start with initialisation in 0^n since here it is easy to see that CLONALG is very efficient.

A. CLONALG on OneMax starting in $\mathbf{0^n}$

We first see that random initialisation in 0^n is very efficient in the case of CLONALG mutation since the algorithms jumps directly into the optimum. The following theorem is very easy to show.

Theorem 7. *Let $x_0^{(CLONALG)} = 0^n$. For all $\rho \in \mathbb{R}^+$ and $b \in \mathbb{N}$,*

$$\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(CLONALG)}\right)\Big) = n$$

holds.

Proof. Recall, that each bit in some bit string $x \in \{0,1\}^n$ is flipped with probability $p(x) = e^{-\rho \cdot v}$ for $v = f(x)/n$. With $x_0 = 0^n$, we have $\textsc{OneMax}(x_0) = 0$. Thus, $p(x_0) = e^0 = 1$, independent of ρ, and we are guaranteed to find the optimum of OneMax after a single mutation. □

B. CLONALG on OneMax starting in $\mathbf{1^{n/2}0^{n/2}}$

In the case of deterministic initialisation in $1^{n/2}0^{n/2}$ things are more complicated since the mutation rate depends on the current function value and thus, is changing over time. However, to make our point, we again only consider the situation directly after initialisation (as we did for CHM, too) and show that with constant probability we make large progress in the very beginning. In this case, RLS needs time to

catch up. We start with a lemma to facilitate our argumentation and show bounds on the probability to deviate from the expectation of binomially distributed variables.

Lemma 1. *Let $X_1, X_2, \ldots, X_k \in \{0,1\}$ be independently identically distributed random variables with $\Pr(X_i = 1) = 1/\sqrt{2k}$ for all $i \in \{1,2,\ldots,k\}$. Then,*

$$\Pr(X \leq \mathrm{E}(X)) \geq \frac{23}{100}$$
$$\Pr\Big(X \geq \mathrm{E}(X) + k^{1/4}\Big) \geq \frac{1}{50}$$

holds for $X := \sum_{i=1}^{k} X_i$ and $k \geq 3$.

Proof. We know that X is binomially distributed with parameters k and $1/\sqrt{2k}$ so that $\mathrm{E}(X) = \sqrt{k/2}$ holds. We start with considering bounds on the probabilities for deviating from the expectation by exactly $k^{1/4}$ and $2k^{1/4}$, respectively. Using

$$\Pr(X = v) = \binom{k}{v} \cdot \left(\frac{1}{\sqrt{2k}}\right)^v \cdot \left(1 - \frac{1}{\sqrt{2k}}\right)^{k-v}$$

and using Stirling's Formula to deal with the Binomial coefficients, one can prove that the following statements hold in the considered setting.

$$\left(\lim_{k\to\infty} \Pr\Big(X = \mathrm{E}(X) - k^{1/4}\Big) \cdot k^{1/4}\right) = \frac{1}{\left(2^{1/4} \cdot \sqrt{\pi} \cdot e^{1/\sqrt{2}}\right)} \approx 0.234$$

$$\left(\lim_{k\to\infty} \Pr\Big(X = \mathrm{E}(X) + 2k^{1/4}\Big) \cdot k^{1/4}\right) = \frac{1}{\left(2^{1/4} \cdot \sqrt{\pi} \cdot e^{2\sqrt{2}}\right)} \approx 0.028$$

From this, it is easy to conclude that

$$\lim_{k\to\infty} \Pr\Big(X = \mathrm{E}(X) - k^{1/4}\Big) \geq \frac{23}{\left(100k^{1/4}\right)}$$
$$\lim_{k\to\infty} \Pr\Big(X = \mathrm{E}(X) + 2k^{1/4}\Big) \geq \frac{1}{\left(50k^{1/4}\right)}$$

holds for sufficiently large k.

We use $p := 1/\sqrt{2k}$ in the following. We know that for $b \in \{0,1,\ldots,k\}$ the probability $\Pr(X = b)$ is maximal for b with $(k+1)p - 1 < b \leq (k+1)p$ [33, Chap. VI.3]. Moreover, $\Pr(X = b')$ is strictly increasing for all $b' < b$ and strictly decreasing for all $b' > b$ [33, Chap. VI.3]. We assume without loss of generality that $\mathrm{E}(X) = kp = \sqrt{k/2} \in \mathbb{N}$ and $k^{1/4} \in \mathbb{N}$. This implies that $\Pr(X = b)$ is maximal for $b = \mathrm{E}(X)$ since $-1 + (k+1)/\sqrt{2k} < \sqrt{k/2} < (k+1)/\sqrt{2k}$ holds.

Using this observation, we can use the above lower bounds on the deviation from the expectation to derive the desired estimate. We know that $\Pr(X = v) \geq 23/\left(100k^{1/4}\right)$ holds for all $v = \mathrm{E}(X) - i$ with $i \in \left\{1,2,\ldots,k^{1/4}\right\}$ and get

$$\Pr(X \leq \mathrm{E}(X)) \geq k^{1/4} \cdot \frac{23}{100k^{1/4}} = \frac{23}{100}.$$

Analogously, we get

$$\Pr\Big(X \geq \mathrm{E}(X) + k^{1/4}\Big) \geq k^{1/4} \cdot \frac{1}{50k^{1/4}} = \frac{1}{50}$$

since we have $\Pr(X = v) \geq 1/\left(50k^{1/4}\right)$ for all $v = \mathrm{E}(X) + k^{1/4} + i$ with $i \in \left\{1,2,\ldots,k^{1/4}\right\}$. □

Using this Lemma and in the same spirit as in Theorem 2 and 4 we can show a lower bound on the expected function value after the first mutation. Based on the results in [9] we restrict our analyses to the setting $\rho = \ln n$ in the following. There it has been pointed out that extremely large and extremely small values for ρ can be problematic. For the intermediate value $\rho = \ln n$ the mutation rate tends to the standard mutation rate $1/n$ and thus, this value seems to be a natural choice.

Theorem 8. *Let $x_0^{(\text{CLONALG})} = 1^{n/2}0^{n/2}$ and $\rho = \ln n$. For all $b \in \mathbb{N}$,*

$$\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\text{CLONALG})}\right)\Big) > \frac{n}{2} + 0.0038n^{1/4}$$

holds.

Proof. After initialisation we have $\textsc{OneMax}\left(x_0^{(\text{CLONALG})}\right) = n/2$. We consider the expected function value after the first mutation.

Recall, that each bit in some bit string $x \in \{0,1\}^n$ is flipped with probability $p(x) = e^{-\rho \cdot v}$ for $v = f(x)/n$. With $x_0 = 1^{n/2}0^{n/2}$, we have $\textsc{OneMax}(x_0) = n/2$ and thus, $p(x_0) = e^{-\ln(n)\cdot(n/2)/n} = n^{-1/2} = 1/\sqrt{n}$.

We first consider the 1-bits in x_0 and assign a random variable $X_i \in \{0,1\}$ with $\Pr(X_i = 1) = p(x_0)$ to each of these 1-bits, indicating if this specific bit is flipped. Then, $M_1 = \sum_{i=1}^{n/2} X_i$ corresponds to the number of 1-bits flipping in the current mutation. Since the X_i are independent and initially, we have $n/2$ 1-bits, we can apply Lemma 1 with $k = n/2$ and get

$$\Pr(M_1 \leq \mathrm{E}(M_1)) \geq \frac{23}{100}$$

for the number of 1-bits flipping to 0. Analogously, we get

$$\Pr\left(M_0 \geq \mathrm{E}(M_0) + \left(\frac{n}{2}\right)^{1/4}\right) \geq \frac{1}{50}$$

for the number of 0-bits flipping to 1. The progress is at least $M_0 - M_1$ with probability at least $(23/100) \cdot (1/50)$. Since $\mathrm{E}(M_0) = \mathrm{E}(M_1)$, we get $M_0 - M_1 \geq (n/2)^{1/4}$. Plugging together what we have, the expected function value after the first mutation is at least

$$\mathrm{E}\Big(\textsc{OneMax}\left(x_1^{(\text{CLONALG})}\right)\Big) \geq \frac{n}{2} + \frac{23}{100} \cdot \frac{1}{50} \cdot \left(\frac{n}{2}\right)^{1/4} > \frac{n}{2} + 0.0038n^{1/4}$$

Since the function value cannot decrease in later iterations, this proves the theorem. □

We now derive an upper bound on the expected function value and show that it does not increase beyond $(1/2 + \varepsilon)n$, $0 < \varepsilon < 1/2$ some constant, within an exponential number of iterations. The main proof idea follows the argumentation

in [9] and shows that the algorithm gets stuck once a sufficiently large number of ones is collected: Due to the very large mutation probability, the mutation operator is then likely to flip more 1-bits to 0-bits than 0-bits to 1-bits.

Theorem 9. *Let* $x_0^{(CLONALG)} = 1^{n/2}0^{n/2}$ *and* $\rho = \ln n$. *For all* $b \in \mathbb{N}_0$ *with* $b \leq e^{dn^{1/2-\varepsilon}}$, $0 < \varepsilon < 1/2$ *constant,* $d > 0$ *constant and sufficiently small,*

$$\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(CLONALG)}\right)\Big) \leq \left(\frac{1}{2}+\varepsilon\right)n + o(1)$$

holds. We even have $\textsc{OneMax}\left(x_b^{(CLONALG)}\right) \leq (1/2+\varepsilon)n$ *with probability* $1 - e^{-\Omega\left(n^{1/2-\varepsilon}\right)}$.

Proof. Similarly to [9] we show that (for sufficiently large n) with probability $1 - e^{-\Omega\left(n^{1/2-\varepsilon}\right)}$ the fitness function value is not increased beyond $(1/2+\varepsilon)n$. This implies that for at least $e^{dn^{1/2-\varepsilon}}$ iterations, the fitness function value is not increased beyond $(1/2+\varepsilon)n$ with probability $1 - e^{-\Omega\left(n^{1/2-\varepsilon}\right)}$ and we get

$$\begin{aligned}
&\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CLONALG})}\right)\Big)\\
&\leq \left(1 - e^{-\Omega\left(n^{1/2-\varepsilon}\right)}\right)\cdot\left(\frac{1}{2}+\varepsilon\right)n + e^{-\Omega\left(n^{1/2-\varepsilon}\right)}\cdot n\\
&\leq \left(\frac{1}{2}+\varepsilon\right)n + o(1)
\end{aligned}$$

We are left with the proof of the probability to increase the function value beyond $(1/2+\varepsilon)n$. Initially we have $\textsc{OneMax}\left(x_b^{(\mathrm{CLONALG})}\right) = n/2$. We observe that the mutation rate is monotonically decreasing in the fitness and thus, it is maximal after initialisation. More precisely, we have $p(x_b) \leq 1/\sqrt{n}$ for all $b \geq 0$. Thus, the expected number of flipping bits is at most $\sqrt{n}$. Due to Chernoff bounds [31] the probability that more than $2\sqrt{n}$ bits flip is $e^{-\Omega\left(\sqrt{n}\right)}$, implying that with high probability we do not make larger 'jumps' towards the optimum. This implies that with probability at least $1 - e^{-\Omega\left(\sqrt{n}\right)}$ we reach an interval $(1/2+\gamma')n \leq \textsc{OneMax}\left(x_b^{(\mathrm{CLONALG})}\right) \leq (1/2+\gamma'')n$, $0 < \gamma' < \gamma'' < \varepsilon < 1/2$ constant, before reaching the optimum.

Consider some iteration with $\textsc{OneMax}\left(x_b^{(\mathrm{CLONALG})}\right) = (1/2+\gamma)n$, $\gamma' < \gamma < \gamma''$ constant, and let M_0 and M_1 denote the number of flipping 0- and 1-bits, respectively. Due to Chernoff bounds, the probability that more than $(1+\delta)\mathrm{E}(M_0)$ or less than $(1-\delta)\mathrm{E}(M_1)$, $0 < \delta < 2\gamma$ constant, many 0- or 1-bits flip is at most $e^{-\Omega(\mathrm{E}(M_0))} = e^{-\Omega\left(n^{1/2-\gamma}\right)}$. Assuming that that many bits flip we still do not see a successful mutation:

$$\begin{aligned}
&M_0 - M_1\\
&\leq (1+\delta)\mathrm{E}(M_0) - (1-\delta)\mathrm{E}(M_1)\\
&= (1+\delta)\cdot\left(\frac{1}{2}-\gamma\right)n\cdot\frac{1}{n^{1/2+\gamma}}\\
&\qquad - (1-\delta)\cdot\left(\frac{1}{2}+\gamma\right)n\cdot\frac{1}{n^{1/2+\gamma}}\\
&= (1+\delta)\left(\frac{1}{2}-\gamma\right)n^{1/2-\gamma} - (1-\delta)\left(\frac{1}{2}+\gamma\right)n^{1/2-\gamma}\\
&= n^{1/2-\gamma}\cdot(\delta - 2\gamma) < 0
\end{aligned}$$

Since $\gamma < \varepsilon$, this yields that with probability $e^{-\Omega\left(n^{1/2-\varepsilon}\right)}$ we do not improve beyond $(1/2+\varepsilon)n$ within $e^{dn^{1/2-\varepsilon}}$ iterations. □

We summarise our results in the following corollary. Note that we restrict our attention to polynomial budgets, only, since larger budgets are not of practical interest.

Corollary 3. *Let* $x_0^{(CLONALG)} = x_0^{(RLS)} = 1^{n/2}0^{n/2}$, $\rho = \ln n$ *and* $b \in \mathbb{N}$.

For all $1 \leq b \leq \left\lfloor 0.0038n^{1/4}\right\rfloor - 1$ *the expected function value after* b *function evaluations is larger for CLONALG than for RLS.*

For all $b \geq \lceil 1.61n\rceil$, $b = n^{O(1)}$, *the expected function value after* b *function evaluations is larger for RLS than for CLONALG.*

Proof. We have

$$\begin{aligned}
&\mathrm{E}\Big(\textsc{OneMax}\left(x_0^{(\mathrm{CLONALG})}\right)\Big)\\
&\qquad = \mathrm{E}\Big(\textsc{OneMax}\left(x_0^{(\mathrm{RLS})}\right)\Big) = n/2
\end{aligned}$$

and

$$\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{CLONALG})}\right)\Big) > \frac{n}{2} + 0.0038n^{1/4}$$

for all $b \in \mathbb{N}$. Since RLS is only flipping exactly one bit per iteration, RLS needs at least $0.0038n^{1/4}$ iterations to gain a progress of $0.0038n^{1/4}$. Thus, after $0.0038n^{1/4} - 1$ iterations, $\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(\mathrm{RLS})}\right)\Big) < n/2 + 0.0038n^{1/4}$ still holds which proves the first part of the claim.

Due to Theorem 9, the expected function value for CLONALG is at most $(1/2+\varepsilon)n + o(1)$ for at least $e^{dn^{1/2-\varepsilon}}$ iterations. We set $\varepsilon = 2/5$ and investigate the number of iterations RLS needs to improve the function value beyond $(1/2+\varepsilon) = 9n/10$. The expected function value for RLS equals $(n/2) + (n/2)\cdot\left(1 - (1-1/n)^b\right)$ (Theorem 1). A straightforward calculation shows that this is larger than $9n/10$ for $b \geq \ln(1/5)/\ln(1-1/n)$. Since $\ln(1-1/n) \approx -1/x$ for large n, we see that $b \geq c\cdot n$, $c > 0$ constant, holds. A more careful calculation shows that $c = 1.61$ is sufficient. Since we restrict ourselves to polynomial budgets, this proves the second claim. □

C. CLONALG on ONEMAX with random starting point

Finally, we consider the case of random initialisation. Compared to the previous section we now have to deal with different possible mutation rates already after initialisation since it is not clear how many 1-bits the initial search point has. However, one can prove that this number is bounded and thus things do not change significantly. Note, that again we only consider the first iteration to make our point.

We start with a lower bound that is of the same order as the one for initialisation in $1^{n/2}0^{n/2}$.

Theorem 10. *Let $x_0^{(CLONALG)}$ be selected uniformly at random and $\rho = \ln n$. For all $b \in \mathbb{N}$,*

$$\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(CLONALG)}\right)\Big) > \frac{n}{2} + 0.0019n^{1/4}$$

holds.

Proof. In comparison to the proof of Theorem 8 we do not have a deterministic initial function value and thus, the mutation rate for the initial search point can vary. Remember that the expected number of initial 1-bits equals $n/2$. We consider different cases to estimate the expected progress in the first mutation. With probability $1/2$ we have at least $n/2$ many 1-bits after initialisation. In this case we have $p(x_0) \leq n^{-1/2} = 1/\sqrt{n}$.

If the initial function value is at least $n/2 + 0.0019n^{1/4}$ there is nothing to show. Thus, to prove the theorem we can further assume that the initial function value is less than $n/2 + 0.0019n^{1/4}$ and thus $p(x_0) \geq n^{-(1/2+n^{-3/4})}$ holds. Note that for $n \to \infty$ this converges to $1/\sqrt{n}$.

Using Lemma 1 with $n/2 \leq k \leq n^{1+2/n^{3/4}}/2$ we show that we gain a progress of $\Omega\big(n^{1/4}\big)$ with probability $\Omega(1)$ and see that the different initialisation does not significantly influence the behaviour of CLONALG. Note that the decreased progress of $0.0019n^{1/4}$ in comparison to Theorem 8 stems from the probability to have at least $n/2$ many 1-bits after initialisation. In all other cases the change in the number of 1-bits cannot be negative and can therefore be estimated by 0. □

For the upper bound we can mostly reuse the proof of Theorem 9. However, we need to additionally ensure that initialisation does not significantly decrease the probability to reach the considered interval, i. e., there exists a b with $(1/2 + \gamma')n \leq \textsc{OneMax}\left(x_b^{(\text{CLONALG})}\right) \leq (1/2 + \gamma'')n$, $0 < \gamma' < \gamma'' < 1/2$ constant.

Theorem 11. *Let $x_0^{(CLONALG)}$ be selected uniformly at random and $\rho = \ln n$. For all $b \in \mathbb{N}_0$ with $b \leq e^{dn^{1/2-\varepsilon}}$, $0 < \varepsilon < 1/2$ constant, $d > 0$ constant and sufficiently small,*

$$\mathrm{E}\Big(\textsc{OneMax}\left(x_b^{(CLONALG)}\right)\Big) \leq \left(\frac{1}{2} + \varepsilon\right) n + o(1)$$

holds. We even have $\textsc{OneMax}\left(x_b^{(CLONALG)}\right) \leq (1/2 + \varepsilon)n$ with probability $1 - e^{-\Omega\left(n^{1/2-\varepsilon}\right)}$.

Proof. The proof follows the line of thought of Theorem 9 and the work done in [9]. By Chernoff bounds, we have $\Pr\Big((1/2 - \varepsilon)n \leq \textsc{OneMax}\left(x_0^{(\text{CLONALG})}\right) \leq (1/2 + \varepsilon)n\Big) = 1 - e^{-\Omega(n)}$. For $\textsc{OneMax}\left(x_0^{(\text{CLONALG})}\right) \geq n/2$, the proof from Theorem 9 carries over. For $\textsc{OneMax}\left(x_0^{(\text{CLONALG})}\right) < n/2$, we can reuse results from Theorem 5 in [9] and see that with probability at least $1 - e^{-\Omega(n)}$ we do not jump beyond $(1/2 + \varepsilon)n$ 1-bits. Analogously to the proof of Theorem 9 we can conclude the above claim. □

In comparison to the proof of Corollary 3 the only change is in the initial function value. Thus, using essentially the same arguments we get the following summarised result in the case of random initialisation.

Corollary 4. *Let $x_0^{(CLONALG)}$ and $x_0^{(RLS)}$ be selected uniformly at random, $\rho = \ln n$ and $b \in \mathbb{N}$.*

For all $1 \leq b \leq \lfloor 0.0019n^{1/4} \rfloor - 1$ the expected function value after b function evaluations is larger for CLONALG than for RLS.

For all $b \geq \lceil 1.61n \rceil$, $b = n^{O(1)}$, the expected function value after b function evaluations is larger for RLS than for CLONALG.

V. Experimental Supplements

The results in Section III and Section IV prove that theorems about the expected optimisation time that indicate that random local search is much more efficient on OneMax than hypermutations are misleading. Depending on initialisation the immune-inspired algorithms can be superior for quite some time and only be outperformed in the long run. While the theorems from Section III and Section IV prove this in an abstract sense it provides us with a more concrete and clearer picture to consider the situation in practice, too. Therefore, we consider the results of experiments in this section.

We consider the three different initialisations considered in the preceding sections, namely deterministic initialisation in 0^n, deterministic initialisation in 1^n0^n, and random initialisation. For each type of initialisation and each of the three algorithms we perform 100 independent runs where we consider some still realistic but not too small value of n, namely $n = 1000$. Our choice of n is mainly motivated by the observation that for much smaller values users would probably wait sufficiently long for RLS to find the optimum of OneMax. For $n = 1000$ its expected optimisation time is $\approx n \ln(n) \approx 6908$. We consider roughly half that many steps and stop each run after 3500 function evaluations. The rationale behind this lies in the nature of the fixed budget perspective. It has been pointed out that for budgets close to the expected optimisation time one is rather interested in the probability that the optimum has already been found [27]. However, the number of function evaluations considered should be as large as possible to provide meaningful results. Setting the number of steps to half the expected optimisation seems to be a natural choice.

In Figure 2 we plot all average observed function values for each step $t \in \{0, 1, \ldots, 3500\}$ together with the observed standard deviation. This demonstrates that the function values are rather concentrated around the averages for all algorithms and starting points. Therefore, we present only the averages in the other diagrams to make them less cluttered. It is noteworthy that the standard deviation is much larger for CHM than it is for RLS and CLONALG. This is not difficult to understand. For RLS, the progress in each step is either 0 or 1 since exactly one bit is flipped. The process of RLS on OneMax is identical with the well known coupon collecting process [31] and it is well known that it is very much concentrated. For CLONALG, the progress in one step is determined by a large number of random experiments, one for each bit. Therefore, it is also very much concentrated around its mean. For CHM things are different. The progress in one step is determined by only two random experiments (one for the position and the other for the

length of the block) and can be any integer between 0 and n. Thus, the variance is quite large as is evident from the much larger standard deviation.

For deterministic initialisation in 0^n, Corollary 1 states that CHM outperforms RLS in the beginning of the run and is outperformed by it later on. Theorem 7 states that CLONALG will find the optimum in the first mutation and thus will be always best among the three algorithms. We see in Figure 3 that this is the case. The initial advantage of CHM over RLS is so large that it takes RLS almost 3000 function evaluations to catch up. In order to investigate the significance of these results we have performed Wilcoxon signed rank tests for each pair of algorithms and each possible budget. Due to the large number of tests, we perform Holm-Bonferroni correction and depict the resulting p-values in Figure 3 along with the standard confidence level of 0.05. We see that the differences are significant at confidence level 0.05 after initialisation and for RLS and CHM around the point of time when the corresponding curves of the expected function values intersect.

For deterministic initialisation in $1^{n/2}0^{n/2}$, Corollary 2 states that CHM outperforms RLS in the beginning of the run and is outperformed by it later on, however to a clearly lesser degree than when initialising in 0^n. Corollary 3 states similar behaviour for CLONALG in comparison to RLS but for a much shorter period of time. We see in Figure 4 that this is the case. CHM and CLONALG both perform considerably better than RLS in the beginning. RLS needs about 500 function evaluations to catch up with CLONALG and about 1800 function evaluations to catch up with CHM. We again perform statistical tests and depict the results in Figure 4. We see that the differences are significant at confidence level 0.05 after initialisation and around the points of time when two curves of the corresponding expected function values intersect.

For random initialisation, Theorem 6 states that CHM is always worse than RLS. Corollary 4 states that CLONALG performs not much different from initialisation in $1^{n/2}0^{n/2}$, i. e., initially outperforming RLS and being worse later on. We see in Figure 5 that for CLONALG this is the case. While CHM is very slow as predicted it performs surprisingly well in the first three or four steps. This is due to the large variance where a single lucky mutation increasing the number of 1-bits by some number D (say $D \in \{4, 5, \ldots, 9\}$) leads to an advantage that RLS cannot catch up with in less than D steps. As before we perform statistical tests (Figure 5) and see that the differences are significant at confidence level 0.05 except for some short period after initialisation and around the points of time when two curves of the corresponding expected function values intersect.

VI. From Analysis to Design

We have seen that depending on the choice of the initial starting point hypermutations can make far more progress than random local search. We have also seen that this huge benefit in the beginning quickly decreases while optimisation progresses and that, in the end, random local search outperforms hypermutations. This motivates a hybrid approach where one uses hypermutations in the beginning of the search when those promise to lead to larger progress and switch to single bit mutations (i. e., random local search) later.

Designing such hybrid or adaptive algorithms is a very active field across different classes of (nature-inspired) search heuristics. One prominent example are memetic algorithms (see e. g., [34] for a recent overview), a hybridisation of evolutionary algorithms and local search. A recent development in this area considers adaptive memetic algorithms [35] [36] where a local search operator is adaptively selected from a pre-defined set of different such operators. There is also some similarity of this technique to more classical methods such as variable neighbourhood search [37]. Another emerging research direction are hyper-heuristics [38]–[40], which aim at automating the design of more powerful heuristic methods by selecting, combining and adapting several simpler heuristics.

In this section we investigate this idea for somatic contiguous hypermutations, explore the point of time for the switch and empirically evaluate the approach, and compare it with an adaptive approach based on the same idea but exploiting problem-specific knowledge. Since exploiting problem-specific knowledge is a significant advantage we consider this adaptive approach not so much as competition but as a baseline for comparison, an implementation of an ideal case that explores what can be achieved.

In order to estimate the progress a somatic contiguous hypermutation can deliver we concentrate on the length of the longest block of 0-bits. By ignoring other ways of increasing the number of 1-bits our estimate becomes pessimistic.

Lemma 2. *If a bit string contains a block of s consecutive 0-bits with $s \leq n/3$, then the expected increase in the number of 1-bits after application of somatic contiguous hypermutation and selection is bounded below by $s^3/(2n^2)$.*

Proof. We only consider mutations that are at least partly within the block of s 0-bits. First we consider mutations where the starting position is within this block and the length of the mutating block of bits is chosen such that the end is within the block of s 0-bits, too. Second, we consider mutations where the starting position is within this block but the length is so large that the end of the mutating block is outside this block. Finally, we consider mutations that start outside the block of s 0-bits but end in it. We pessimistically assume that outside of the block of s 0-bits we only have 1-bits. This implies that the second and third case are symmetric and we obtain

$$\left(\sum_{p=0}^{s-1}\sum_{l=1}^{s-p}\frac{l}{n(n+1)}\right)+2\sum_{p=0}^{s-1}\sum_{l=s-p+1}^{2(s-p)-1}\frac{s-p-(l-(s-p))}{n(n+1)} = \frac{s^3+s^2}{2n^2+n} \geq \frac{s^3}{2n^2}$$

as lower bound. □

At least if $s^3/(2n^2)$ is larger than $1 - s/n$ we can expect somatic contiguous hypermutations to be more profitable than single bit mutations. Thus, we should apply them at least as long as $s \geq (2n^2)^{1/3}$ holds.

Consider the situation where we start with 0^n. Each somatic contiguous hypermutations has the potential to introduce two

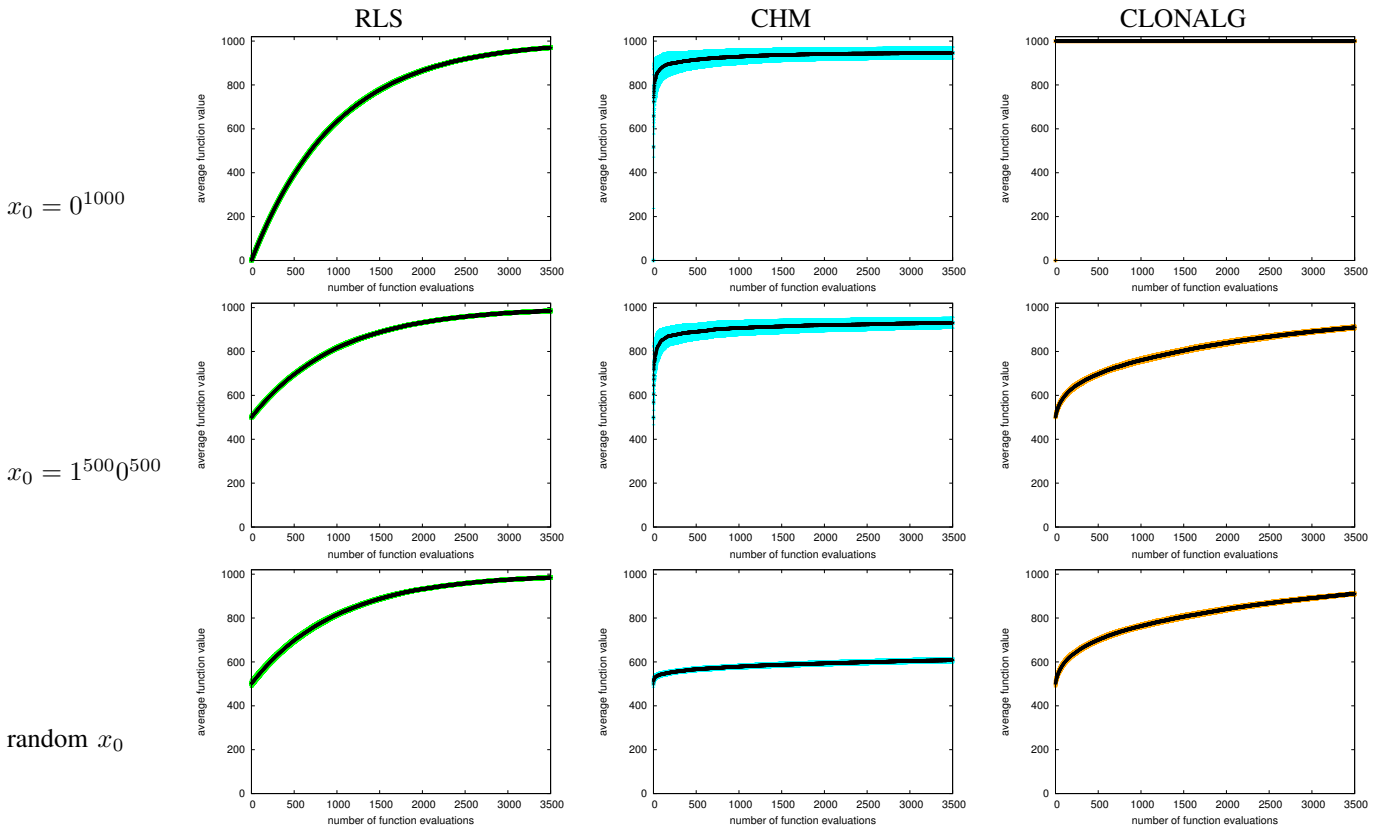

Fig. 2. Average function values together with standard deviations on ONEMAX for 1000 bits, for initialisation in 0^{1000} (top row), $1^{500}0^{500}$ (middle row), and random initialisation (bottom row), for the algorithms RLS (left column), CHM (middle column), and CLONALG (right column), plotted over the number of function evaluations. Since the standard deviations are very small they appear jointly with the averages similar to a single thick line.

separations into the bit string such that the separations cut the bit string into blocks and each block is either a complete block of 0-bits or a complete block of 1-bits. This implies that the number of such blocks after t steps is bounded above by $2t$. (Note that the number is not $2t+1$ since the mutations wrap around.) Thus, the length of a longest block is bounded below by $n/(2t)$. We assume that there is a block of this length that is still a 0-block. Thus, we use $n/(2t)$ as lower bound on the length of a longest block of 0-bits. We want this length to be bounded below by $(2n)^{1/3}$ to see an advantage for somatic contiguous hypermutations. Thus, when starting with 0^n we will use somatic contiguous hypermutations for the first $\lceil n^{1/3}/(2 \cdot 2^{1/3}) \rceil$ steps. When starting in $1^{n/2}0^{n/2}$ not much is changed since already the first somatic contiguous hypermutations leads from 0^n to a bit string where our crude estimate for the length of a longest blocks of 0-bits is $n/4$. Thus, in the following experiments we use somatic contiguous hypermutations for the first $\lceil n^{1/3}/(2 \cdot 2^{1/3}) \rceil$ steps when starting with 0^n or $1^{n/2}0^{n/2}$. Note that the actual block size may be much larger because our simple calculations assumes that every step introduces new blocks. In reality only hypermutations that lead to search points with at least equal fitness do this.

In the unlikely case that we overestimate the length of a longest block of 0-bits we may 'waste' steps by performing somatic contiguous hypermutations in a situation where single bit flips are expected to be more profitable. The number of steps we may do this is $O(n^{1/3})$. The gain we expect by starting with somatic contiguous hypermutations and switching to single bit flips in comparison to doing only random local search is $\Theta(n)$. (This follows directly from the results about the coupon collector scenario when comparing the situation of a start without any coupons with the situation where initially a linear number of coupons is given.) Thus, the potential waste of $O(n^{1/3})$ should be negligible. We investigate this empirically by comparing the simple heuristic that switches between the two mutation operators with an adaptive variant that selects this point of time a bit more cleverly.

We use the same heuristic to devise this adaptive algorithm. We scan the current bit string to determine the current number of 0-bits z and the length of a longest block of 0-bits s. Using Lemma 2 we apply somatic contiguous hypermutations if $s^3/(2n^2) \geq 1 - z/n$ holds and single bit mutations otherwise. Note that this adaptive variant is a bit more computational expensive than the hybrid variant since it requires the determination of the length of the longest blocks of 0-bits. If this is integrated into the computation of the fitness value the additional computational effort becomes negligible, though.

We report the results of experiments for $n \in \{250, 500, 750, \ldots, 5000\}$. For each value of n and each algorithm we perform 100 independent runs and use boxplots to visualise the results. We perform experiments using the three different starting points we have used throughout the paper. This does not only have the advantage to match the

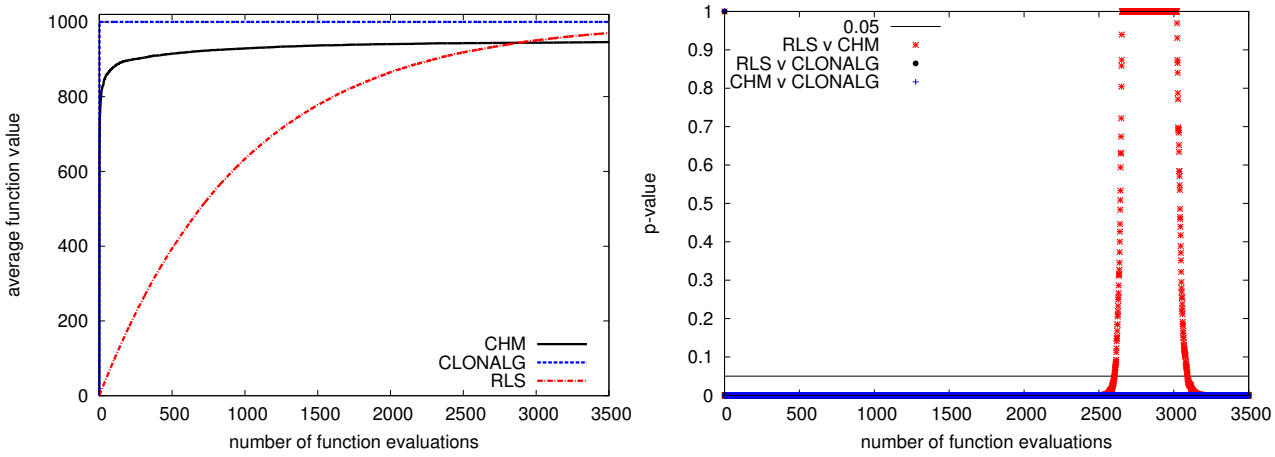

Fig. 3. Average function values on ONEMAX for 1000 bits (left) and p-values of the Wilcoxon tests after Holm-Bonferroni correction (right), for initialisation in 0^{1000} and all three algorithms, plotted over the number of function evaluations.

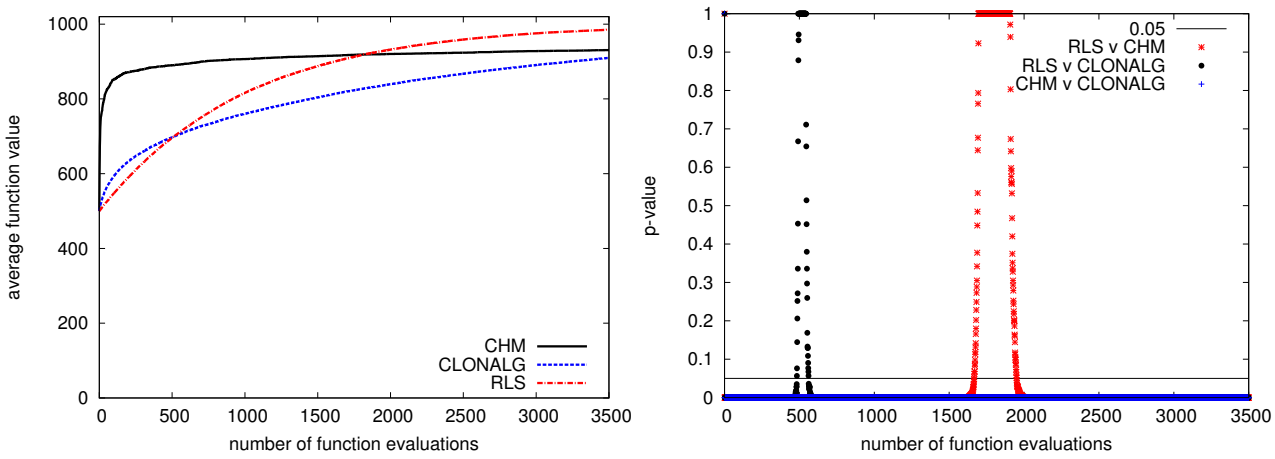

Fig. 4. Average function values on ONEMAX for 1000 bits (left) and p-values of the Wilcoxon tests after Holm-Bonferroni correction (right), for initialisation in $1^{500}0^{500}$ and all three algorithms, plotted over the number of function evaluations.

discussion in the previous sections. It can also match the situation in applications. E. g., in combinatorial optimisation it is common practice not to start with a random search point but with a trivial feasible solution (which often is 0^n or 1^n) [41]. Moreover, fitness landscapes may have the property that the population will almost surely pass through a specific point in the search space. This is very similar to starting in such a specific point. See work by Jansen, De Jong and Wegener [42] for an example and note that Definition 7 in that paper can be used as a general construction method.

In Figure 6 we see the results for the three different choices of the initial bit string we consider. We see that the hybrid algorithm is clearly faster than random local search. We know that there is no asymptotical advantage but the advantage that we expect to be in the order of $\Theta(n)$ (for a runtime that is $n \ln(n) \pm \Theta(n)$) is clearly visible and would be noticeable in practice. We see that the adaptive variant is even faster, outperforming the hybrid approach. This holds for both deterministic initialisations $x_0 = 0^n$ and $x_0 = 1^{n/2}0^{n/2}$. When the initial bit string is chosen uniformly at random we do not expect any advantage when using somatic contiguous hypermutations in the beginning since there are no long blocks of 0-bits. This is also confirmed empirically since we see hardly any difference in the performance of random local search, the hybrid algorithm and the adaptive algorithm in the right column of Figure 6. We note, however, that the initial use of somatic contiguous hypermutations does not constitute a significant waste that would be visible as clearly inferior performance.

To give a clearer idea of the gain in performance we fit the function $n \ln(n) + cn$ to the mean values obtained in the 100 runs for each of the three different choices of the initial search point x_0 and each of the three algorithms. The results can be seen in Table I. As before, we additionally perform Wilcoxon signed rank tests to investigate the significance of the observed differences. The results are shown in Table II and indicate that results for initialisation in 0^n and $1^{n/2}0^{n/2}$ are significant at confidence level 0.05.

We see that the hybrid and the adaptive algorithms have clear advantage over random local search for initialisation in 0^n and $1^{n/2}0^{n/2}$. We can quantify the advantage of the hybrid algorithm as $1.6n$ for $x_0 = 0^n$ and $1.3n$ for $x = 1^{n/2}0^{n/2}$. The adaptive algorithm is about $0.7n$ faster than this in both cases. Again, when the initial search point is selected randomly there

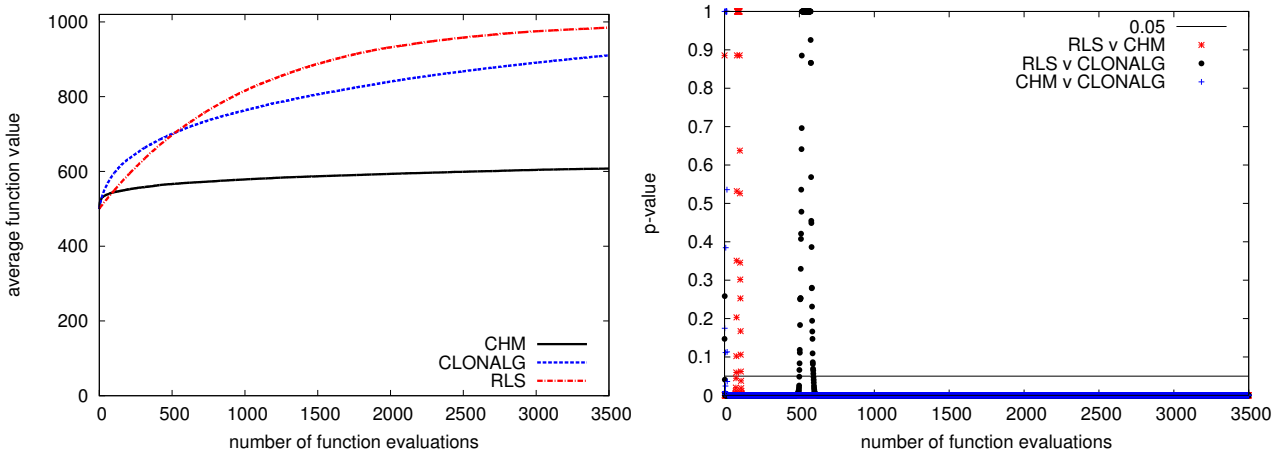

Fig. 5. Average function values on ONEMAX for 1000 bits (left) and p-values of the Wilcoxon tests after Holm-Bonferroni correction (right), for random initialisation and all three algorithms, plotted over the number of function evaluations.

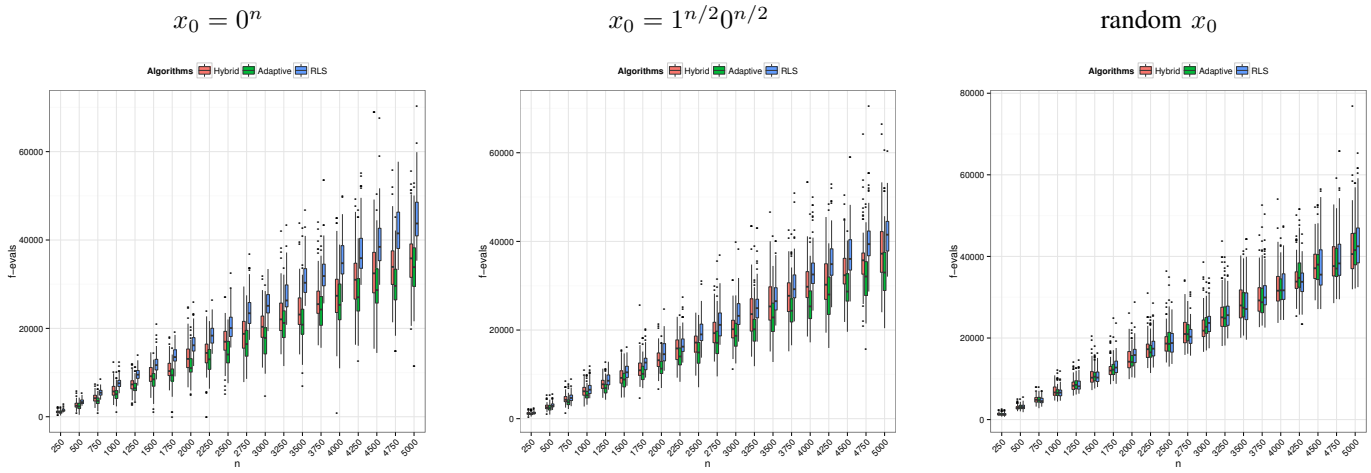

Fig. 6. Boxplots of number of function evaluations needed to optimise ONEMAX for different initial bit strings x_0: for $x_0 = 0^n$ in the left column, for $x_0 = 1^{n/2}0^{n/2}$ in the middle column, for random x_0 in the right column. The boxplots annotated 'RLS' refer to random local search, 'hybrid' refers to the hybrid variant that switches from somatic contiguous hypermutations to local search after the first $\lceil n^{1/3}/(2 \cdot 2^{1/3}) \rceil$ steps, 'adaptive' refers to the adaptive variant.

	$x_0 = 0^n$	$x_0 = 1^{n/2}0^{n/2}$	random x_0
RLS	0.3	-0.3	-0.3
hybrid	-1.3	-1.0	-0.4
adaptive	-2.0	-1.8	-0.3

TABLE I
VALUES OF c OBTAINED WHEN FITTING THE FUNCTION $n \ln(n) + cn$ TO THE MEAN VALUES OBTAINED IN THE 100 RUNS FOR EACH OF THE THREE DIFFERENT CHOICES OF THE INITIAL SEARCH POINT x_0 AND EACH OF THE THREE ALGORITHMS.

	$x_0 = 0^n$	$x_0 = 1^{n/2}0^{n/2}$	random x_0
RLS v hybrid	$< 2.2e$-16	$< 2.2e$-16	0.1116
RLS v adaptive	$< 2.2e$-16	$< 2.2e$-16	0.2505
hybrid v adaptive	$< 2.2e$-16	$< 2.2e$-16	0.3128

TABLE II
p-VALUES OF WILCOXON SIGNED RANK TESTS.

is hardly any difference between the three algorithms.

Since the adaptive algorithm outperforms the hybrid algorithm it is interesting to find out when it switches from somatic contiguous hypermutations to local search. We display in Figure 7 the number of steps the adaptive and hybrid algorithm make use of somatic contiguous hypermutations. For the hybrid algorithm this number is always equal to $\lceil n^{1/3}/(2 \cdot 2^{1/3}) \rceil$ (and, consequently, the boxplots degenerate to a line representing this value). For the adaptive algorithm this number depends on the initialisation. We see that the adaptive algorithm uses somatic contiguous hypermutations much longer than the hybrid algorithm for the two deterministic initialisation variants. However, when the search point is randomly chosen it does not make use of somatic contiguous hypermutations at all which is now less than the hybrid variant.

VII. CONCLUSIONS

Recently, the traditional analytical perspective for randomised search heuristics in the context of optimisation has been challenged and an alternative model has been proposed, fixed budget computations. It has been argued that it offers a perspective that is more in line with the way randomised search heuristics like evolutionary algorithms and artificial immune systems are actually applied. Comparing random local search

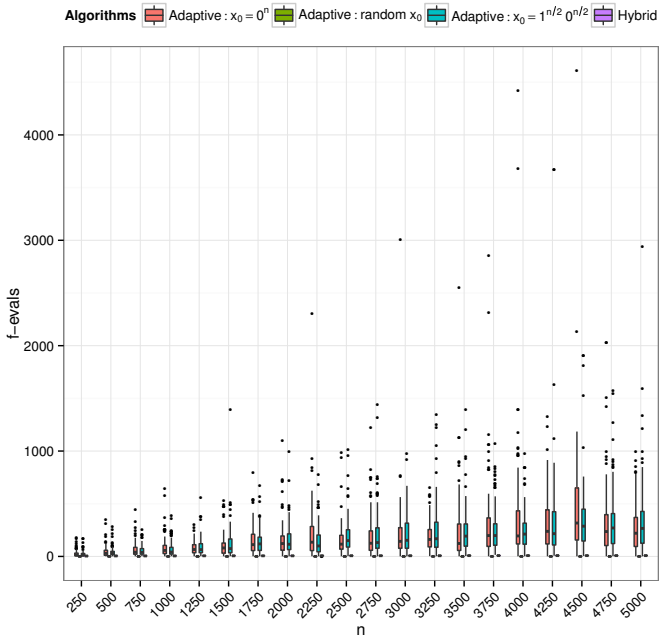

Fig. 7. Boxplots of the number of steps the adaptive and hybrid algorithms make use of somatic contiguous hypermutations. For the hybrid algorithm this number is always $\lceil n^{1/3}/(2 \cdot 2^{1/3}) \rceil$ regardless of the choice of the initial bit string. For the adaptive algorithm this number is a random variable and depends on the choice of the initial bit string.

with two hypermutation operators, one from CLONALG and the other from the B-cell algorithm, we have proved that this novel perspective can lead to radically different findings.

Using one of the best known example functions, ONEMAX, we have compared the fixed budget performance of these three mutation operators embedded in a minimalistic algorithmic framework. It was known before that under the traditional perspective of expected optimisation time RLS outperforms CHM clearly (beating it by a factor of $\Theta(n)$ where n is the length of the bit strings) and that RLS outperforms CLONALG by far (polynomial vs. exponential expected optimisation time). We have demonstrated analytically and empirically that the perspective of fixed budget computation leads to very different results.

We have considered three different choices of the starting point, namely deterministic start in the all zero bit string, in the bit string starting with half the bits being 1-bits and the second half of the bits being 0-bits, and with a bit string selected uniformly at random. CLONALG outperforms RLS with all three starting points, at least at the beginning of the run. If the start point is different from 0^n, however, it is eventually overtaken later in the run. CHM has a better start than RLS with the two deterministic starting points, also being eventually overtaken later in the run. With the random starting point its performance is poor. We see that using the fixed budget perspective we obtain a much more balanced and differentiated picture of the performance of algorithms in comparison to results based on expected optimisation time. It depends on the computational budget one is willing to allocate which algorithm performs best.

Our results show that the traditional perspective of expected optimisation time has its limitations: It may be unable to explain observed good performance that is due to limiting the length of runs. Since in practice one always considers runs of limited length this restriction is serious. Therefore, we have demonstrated that the perspective of fixed budget computations provides valuable information and additional insights.

We have turned our theoretical results into practical ideas by using a simple idea based on the gained insights to devise a hybrid algorithm that applies somatic contiguous hypermutations in the beginning when they can be expected to be more beneficial and switching to local search when somatic contiguous hypermutations start becoming too slow. The number of steps somatic contiguous hypermutations are used is based on our theoretical findings. We demonstrate empirically that this hybrid algorithm exhibits noticeable improvements in performance in comparison with random local search for initialisation in 0^n and $1^{n/2}0^{n/2}$ (not for random initialisation). To evaluate the effectiveness of this simple hybrid approach we compare it with a more sophisticated adaptive approach that determines the number of steps somatic contiguous hypermutations are used based on the expected progress of the two different mutation operators. Experiments reveal that this more complicated algorithm yields no significant advantage which validates the effectiveness of the simple hybrid approach.

Clearly, our results are restricted to one very simple toy problem and simplistic algorithms concentrating on the choice of the mutation operator, only. Results for more example problems and more complete algorithms are needed. We hope that the introduction of fixed budget computations as analytical perspective to the field of artificial immune systems helps practitioners to appreciate theoretical results as practically relevant. Moreover, we hope that demonstrating the feasibility of fixed budget analysis in the context of artificial immune systems motivates theoreticians to perform more advanced and ambitious analyses.

REFERENCES

[1] M. Read, P. S. Andrews, and J. Timmis, "An introduction to artificial immune systems," in *Handbook of Natural Computing*, G. Rozenberg, T. Bäck, and J. N. Kok, Eds. Springer, 2012, pp. 1575–1598.
[2] K. A. De Jong, *Evolutionary Computation: A Unified Approach*. MIT Press, 2006.
[3] M. Dorigo and T. Stützle, *Ant Colony Optimization*. MIT Press, 2004.
[4] K. Dowsland and J. M. Thompson, "Simulated annealing," in *Handbook of Natural Computing*, G. Rozenberg, T. Bäck, and J. N. Kok, Eds. Springer, 2012, pp. 1623–1656.
[5] F. W. Glover and M. Laguna, *Tabu Search*. Springer, 1998.
[6] W. Michiels, E. Aarts, and J. Korst, *Theoretical Aspects of Local Search*. Springer, 2007.
[7] O. Mersmann, M. Preuss, and H. Trautmann, "Benchmarking evolutionary algorithms: Towards exploratory landscape analysis," in *Proceedings of the 11th International Conference on Parallel Problem Solving From Nature (PPSN XI)*. Springer, 2010, pp. 73–82, LNCS 6239.
[8] J. Timmis, A. Hone, T. Stibor, and E. Clark, "Theoretical advances in artificial immune systems," *Theoretical Computer Science*, vol. 403, no. 1, pp. 11–32, 2008.
[9] C. Zarges, "Rigorous runtime analysis of inversely fitness proportional mutation rates," in *Proceedings of the 10th International Conference on Parallel Problem Solving from Nature (PPSN X)*. Springer, 2008, pp. 112–122, LNCS 5199.

[10] ——, "On the utility of the population size for inversely fitness proportional mutation rates," in *Proceedings of the 10th ACM SIGEVO Conference on Foundations of Genetic Algorithms (FOGA 2009)*. ACM Press, 2009, pp. 39–46.
[11] T. Jansen and C. Zarges, "Analyzing different variants of immune inspired somatic contiguous hypermutations," *Theoretical Computer Science*, vol. 412, no. 6, pp. 517–533, 2011.
[12] ——, "Variation in artificial immune systems: Hypermutations with mutation potential," in *Proceedings of the 10th International Conference on Artificial Immune Systems (ICARIS 2011)*, ser. Lecture Notes in Computer Science, vol. 6825. Springer, 2011, pp. 132–145.
[13] ——, "On the role of age diversity for effective aging operators," *Evolutionary Intelligence*, vol. 4, no. 2, pp. 99–125, 2011.
[14] ——, "On benefits and drawbacks of aging strategies for randomized search heuristics," *Theoretical Computer Science*, vol. 412, no. 6, pp. 543–559, 2011.
[15] P. S. Oliveto and D. Sudholt, "On the runtime analysis of stochastic ageing mechanisms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2014)*. ACM Press, 2014, pp. 113–120.
[16] T. Jansen, P. S. Oliveto, and C. Zarges, "On the analysis of the immune-inspired b-cell algorithm for the vertex cover problem," in *Proceedings of the 10th International Conference on Artificial Immune Systems (ICARIS 2011)*, ser. Lecture Notes in Computer Science, vol. 6825. Springer, 2011, pp. 117–131.
[17] T. Jansen and C. Zarges, "Computing longest common subsequences with the b-cell algorithm," in *Proceedings of the 11th International Conference on Artificial Immune Systems (ICARIS 2012)*, ser. Lecture Notes in Computer Science, vol. 7597. Springer, 2012, pp. 111–124.
[18] ——, "Evolutionary algorithms and artificial immune systems on a bi-stable dynamic optimisation problem," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2014)*. ACM Press, 2014, pp. 975–982.
[19] M. Elberfeld and J. Textor, "Negative selection algorithms on strings with efficient training and linear-time classification," *Theoretical Computer Science*, vol. 412, no. 6, pp. 534–542, 2011.
[20] J. Textor, "Efficient negative selection algorithms by sampling and approximate counting," in *Proceedings of the 12th International Conference on Parallel Problem Solving from Nature (PPSN XII), Part I*, ser. Lecture Notes in Computer Science, vol. 7491. Springer, 2012, pp. 32–41.
[21] H. Bernardino and H. Barbosa, "Artificial immune systems for optimization," in *Nature-Inspired Algorithms for Optimisation*, R. Chiong, Ed. Springer, 2009, pp. 389–411.
[22] L. N. de Castro and F. J. Von Zuben, "Learning and optimization using the clonal selection principle," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 3, pp. 239–251, 2002.
[23] L. N. de Castro and J. Timmis, "An artificial immune network for multimodal function optimization," in *Proceedings of the 4th IEEE Congress on Evolutionary Computation (CEC 2002)*, 2002, pp. 699–704.
[24] J. Kelsey and J. Timmis, "Immune inspired somatic contiguous hypermutation for function optimisation," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*. Springer, 2003, pp. 207–218, LNCS 2723.
[25] T. Jansen, *Analyzing Evolutionary Algorithms. The Computer Science Perspective*. Springer, 2013.
[26] T. Jansen and C. Zarges, "Fixed budget computations: A different perspective on run time analysis," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2012)*. ACM Press, 2012, pp. 1325–1332.
[27] ——, "Performance analysis of randomised search heuristics operating with a fixed budget," *Theoretical Computer Science*, vol. 545, pp. 39–58, 2014.
[28] B. Doerr, T. Jansen, C. Witt, and C. Zarges, "A method to derive fixed budget results from expected optimisation times," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2013)*. ACM Press, 2013, pp. 1581–1588.
[29] S. Nallaperuma, F. Neumann, and D. Sudholt, "A fixed budget analysis of randomized search heuristics for the traveling salesperson problem," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2014)*. ACM Press, 2014, pp. 807–814.
[30] B. Doerr, M. Fouz, and C. Witt, "Sharp bounds by probability-generating functions and variable drift," in *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference (GECCO 2011)*. ACM, 2011, pp. 2083–2090.
[31] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1997.
[32] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+1) evolutionary algorithm," *Theoretical Computer Science*, vol. 276, pp. 51–81, 2002.
[33] W. Feller, *An Introduction to Probability Theory and Its Applications*, 3rd ed. John Wiley & Sons, 1968, vol. 1.
[34] F. Neri, C. Cotta, and P. Moscato, Eds., *Handbook of Memetic Algorithms*. Springer, 2012.
[35] J. E. Smith, "Self-adaptative and coevolving memetic algorithms," in *Handbook of Memetic Algorithms*, F. Neri, C. Cotta, and P. Moscato, Eds. Springer, 2012, pp. 167–188.
[36] Y.-S. Ong, M.-H. Lim, N. Zhu, and K. W. Wong, "Classification of adaptive memetic algorithms: a comparative study," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 36, no. 1, pp. 141–152, 2006.
[37] P. Hansen, N. Mladenović, J. Brimberg, and J. A. M. Pérez, "Variable neighborhood search," in *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds. Springer, 2010, pp. 61–86.
[38] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," in *Handbook of Metaheuristics*, F. Glover and G. A. Kochenberger, Eds. Springer, 2003, pp. 457–474.
[39] E. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. Woodward, "A classification of hyper-heuristic approaches," in *Handbook of Metaheuristics*, M. Gendreau and J.-Y. Potvin, Eds. Springer, 2010, pp. 449–468.
[40] E. K. Burke, M. Gendreau, M. R. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.
[41] I. Wegener, "Simulated annealing beats Metropolis in combinatorial optimization," in *Proceedings of the 32nd International Colloquium on Automata, Languages and Programming (ICALP 2005)*. Springer, 2005, pp. 589–601.
[42] T. Jansen, K. De Jong, and I. Wegener, "On the choice of the offspring population size in evolutionary algorithms," *Evolutionary Computation*, vol. 13, no. 4, pp. 413–440, 2005.

Thomas Jansen received the Diploma and Ph. D. degrees in Computer Science from the Technical University Dortmund, Germany, in 1996 and 2000, respectively. His Ph. D. thesis on the theoretical analysis of evolutionary algorithms was awarded the University Best Dissertation Award.

He was Post-doc researcher at the George Mason University 2001–2002, Juniorprofessor for Computational Intelligence at the Technical University of Dortmund 2002–2009, Stokes Lecturer at the University College Cork, Ireland, 2009–2012 and is Senior Lecturer at Aberystwyth University, UK, since 2013. He has authored 20 journal papers, 42 conference articles, 6 book chapters and one text book on Analyzing Evolutionary Algorithms. He is Senior Member of the ACM and associate editor of *Evolutionary Computation* and *Artificial Intelligence*.

Christine Zarges studied Computer Science at the TU Dortmund, Germany, and obtained her Diploma (2007) and PhD (2011, Theoretical Foundations of Artificial Immune Systems) there. Her dissertation was awarded the dissertation award of the TU Dortmund. In 2010 she received a Google Anita Borg Memorial Scholarship.

She is a Birmingham Fellow in the School of Computer Science at the University of Birmingham, UK since 2012. Before that she spent 12 months as a visiting PostDoc at the University of Warwick, UK, supported by a scholarship of the German Academic Research Service (DAAD). Her research focuses on the theoretical analysis of randomised algorithms and processes, in particular randomised search heuristics such as artificial immune systems and evolutionary algorithms. She is also interested in computational and theoretical aspects of immunology. She is member of the editorial board of *Evolutionary Computation*.